Intro to UNIX

Advanced Track: Lecture 1

Bernard Zhao

Administrivia

- Welcome to the Advanced Track of the DeCal!
- The DeCal is P/NP (surprise)
- Labs due the week after they're assigned, can turn up to two of them late.
  - Gradescope lab for A1 should be released.
- Lectures and Labs are released every Thurs

Course resources

- Your facilitators!
- All the material (new and old) is online!
- Office hours + demos during the in-person lecture time - 8pm PST in the ocf.io/lab
- Email us or drop by in #decal-general via Discord, Slack, IRC, Matrix, etc.)

Engaging with this lecture

- Follow slides online: ocf.io/decal/slides/a1
- Connect to login server:
  - `$OCF_USERNAME@ssh.ocf.berkeley.edu`
- Ask questions!
  - Out loud
  - On #decal-general
  - Lectures are a lot more fun when you ask questions.

Agenda

1. The Shell
2. Unix
3. FOSS

The Shell

Mental model of understanding the shell

Level 0:

- Shell as a black box. Things get typed in, things happen.
- Basically the level that CS 61A requires.

Level 1:

- Executing programs by name, collect and display results

Level 2:

- Chaining commands together, manipulating output between, and understanding what's happening under the hood.

Demo

Shell Commands

- Navigation: `ls`, `cd`
- File Ops: `cp`, `mv`, `rm`, `link`
- Read: `cat`, `less`, `head`, `tail`
- Search: `find`, `grep`, `which`
- Manipulate: `cut`, `sed`, `awk`, `uniq`
- Text Editing: `nano`, `vim`, ~~`emacs`~~
- Retrieve: `curl`, `wget`
- Compress: `tar`, `gzip`, `gunzip`
- Documentation: `man`, `apropos`

(You'll see more in the network lecture)

Control flow stuff:

- Aliases: `alias rm='rm -i'`
- Pipes: `ls -l | wc`
- Variables: `$1-9`, `$?`, `$#`, `$!`, `$$`
- Expansions: `mkdir a/{b,c}`
- Redirecting stdin / stdout: >, >>, <
  - `2 > /dev/null`
  - `input < $file`

Demo

The demo can be found at asciinema.

That was slightly painful. First time trying it out so it’ll be a bit rough.

History and Core Concepts

History of UNIX

- 1960’s: Bell Labs, MIT, GE: Multics
- 1970’s: Bell Labs: UNIX (dmr, Ken Thompson)
- 1980’s: Unix (SysV) derivatives (BSD, Solaris..)
- 1990’s: Linux starts, lawsuits (AT&T v UC), Darwin
- 2000’s: Linux takes over
- 20??: Year of the Linux desktop???

Disclaimer

This slide has been added since the resignation of rms from the Free Software Foundation and CSAIL (2019-09-17).

After much controversy (and it not being the first time he has found himself under criticism), rms has resigned from CSAIL at MIT and the Free Software Foundation. This is a good thing that should have happened years ago. He's consistently shown himself no longer to function as a leader, much less one of what should be a leaderless movement.

I'm choosing to leave his writings in these slides because I believe they are intellectually important and the foundation of the free software movement. A longer explanation can be found here. If you find this decision distasteful, then I sincerely apologize but firmly disagree.

GNU and the Free Software Movement

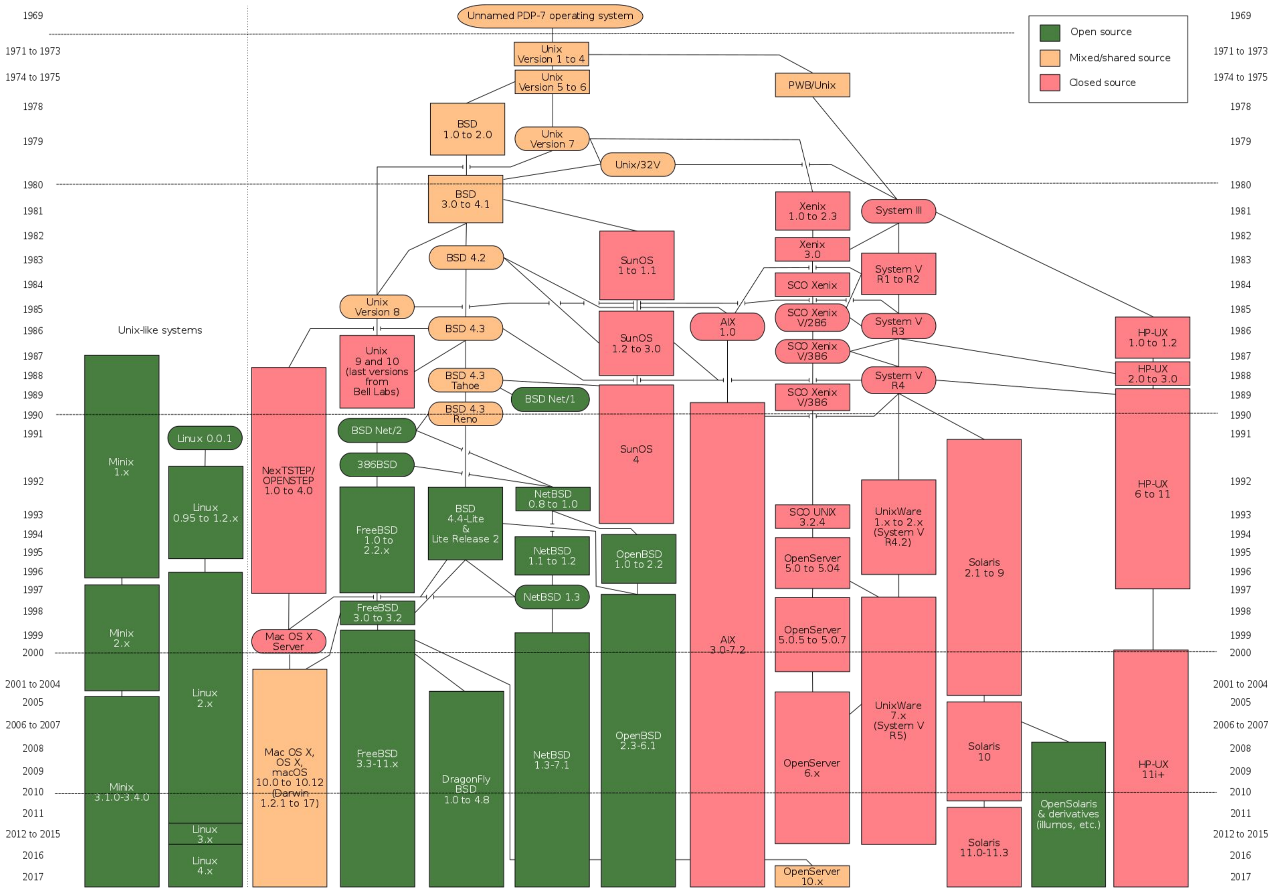
“So that I can continue to use computers without violating my principles, I have decided to put together a sufficient body of free software so that I will be able to get along without any software that is not free.”

Richard Stallman (1983)

https://groups.google.com/forum/#!msg/net.unix-wizards/8twfRPM79u0/1xlglzrWrU0J

Meanwhile at Berkeley...

- A UNIX derivative (BSD) was created at UC Berkeley
- Popular and reliable, a lot of things from BSD are used today
  - vi and Berkeley sockets
- Stuck in legal limbo w/ AT&T
- BSD derivatives still remain popular (MacOS, FreeBSD, OpenBSD)


Open source
Mixed/shared source
Closed source
Unnamed PDP-7 operating system
Unix Version 1 to 4
Unix Version 5 to 6
PWB/Unix
BSD 1.0 to 2.0
Unix Version 7
Unix/32V
BSD 3.0 to 4.1
Xenix 1.0 to 2.3
System III
SunOS 1 to 1.1
Xenix 3.0
BSD 4.2
System V R1 to R2
SCO Xenix
Unix Version 8
SCO Xenix V/286
System V R3
AIX 1.0
BSD 4.3
SunOS 1.2 to 3.0
SCO Xenix V/386
HP-UX 1.0 to 1.2
Unix-like systems
Unix 9 and 10 (last versions from Bell Labs)
BSD 4.3 Tahoe
System V R4
HP-UX 2.0 to 3.0
BSD Net/1
SCO Xenix V/386
BSD 4.3 Reno
Minix 1.x
Linux 0.0.1
NexTSTEP/ OPENSTEP 1.0 to 4.0
BSD Net/2
386BSD
SunOS 4
Linux 0.95 to 1.2.x
FreeBSD 1.0 to 2.2.x
BSD 4.4-Lite & Lite Release 2
NetBSD 0.8 to 1.0
SCO UNIX 3.2.4
UnixWare 1.x to 2.x (System V R4.2)
Solaris 2.1 to 9
HP-UX 6 to 11
NetBSD 1.1 to 1.2
OpenBSD 1.0 to 2.2
OpenServer 5.0 to 5.04
Minix 2.x
NetBSD 1.3
FreeBSD 3.0 to 3.2
Mac OS X Server
OpenServer 5.0.5 to 5.0.7
AIX 3.0-7.2
Linux 2.x
Mac OS X, OS X, macOS 10.0 to 10.12 (Darwin 1.2.1 to 17)
FreeBSD 3.3-11.x
DragonFly BSD 1.0 to 4.8
NetBSD 1.3-7.1
OpenBSD 2.3-6.1
OpenServer 6.x
UnixWare 7.x (System V R5)
Solaris 10
OpenSolaris & derivatives (illumos, etc.)
HP-UX 11i+
Minix 3.1.0-3.4.0
Linux 3.x
Linux 4.x
Solaris 11.0-11.3
OpenServer 10.x
1969
1971 to 1973
1974 to 1975
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001 to 2004
2005
2006 to 2007
2008
2009
2010
2011
2012 to 2015
2016
2017

Unix Design Philosophy

- Minimal, modular software
- Can compose multiple small programs together
- “Do one thing and do it well”
- Against monoliths
  - (Insert systemd flamewar here)
- Everything is a file

Unix Design Philosophy: *Everything* is a file

- Simple, familiar semantics for everything:
  - `read()`, `write()`, `open()`, etc.
- Everything is simply a stream of bytes from a file descriptor
  - Network sockets, pipes, /proc, drivers, etc. etc.
- Familiar filesystem based permissions, access control, etc.

Unix advantages over non-Unix systems

- Worse is better - better to be simple than perfect
- Built to be extensible, introspectable: procfs, strace, dbus
- Bill Joy on implementing TCP/IP: "It's very simple — you read the protocol and write the code."
- Sockets for networking, now copied beyond Unix systems, e.g. WinSock

Free and Open Source Software

What is free software?

(Less free) ← Closed-source | Public domain | Open-source | Copyleft → (More free)

A question of intellectual property: Who owns the code?

- What does it mean to own code?
- With respects to copyright law? Patent law?
- How does distribution work? Modification? Exclusion?
- Is this even the right question to ask? Is “property” the best framing?

The Four Freedoms

- The freedom to run the program as you wish, for any purpose.
- The freedom to study how the program works, and change it so it does your computing as you wish. Access to the source code is a precondition for this.
- The freedom to redistribute copies so you can help others.
- The freedom to distribute copies of your modified versions to others. By doing this you can give the whole community a chance to benefit from your changes. Access to the source code is a precondition for this.

What is free software? by the Free Software Foundation

MIT License

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

[Warranty and liability information]

Permissive (BSD-like) Licensing

- Broadly speaking, it can be interpreted as “Do whatever you want with our code, just don’t sue us”
- Usually, the only responsibilities are with regards to liability, attribution, and patent rights.
- Popular versions include Apache, MIT, and BSD
- Most are super simple in its terms and length

GNU Public License v2 (GPLv2)

Preamble of the GNU Public License v2: “To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.”

∫2(b): “You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

∫3: “You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also [release the corresponding source code].”

Licensing: Copyleft

A “viral” nature (or like a spider plant if you want to be nice): Using GPL code means that your code also has to be free if you want to distribute (includes selling) it.

“I want to make sure that all versions of GNU remain free.”
- Stallman (GNU Manifesto)

“a cancer that attaches itself in an intellectual property sense to everything it touches” - Steve Ballmer

Why FOSS?

- Cost
- Security
- Privacy
- Control
- Collaboration

Tivoization: GPLv2 vs. GPLv3

Some history: When TiVo released its Series 2 v3.2 model in 2006, it came with an anti-modification feature. If the code running on the hardware did not have the appropriate digital signature, the box would either revert the changes or not run at all.

While technically being compliant with the terms of GPLv2, users could not feasibly modify the code and run their own version of TiVo.

In response, the FSF released GPLv3 which prohibited this sort of behavior (sort of). It's fairly contentious and you'll see both versions used frequently. (e.g. Linux is released under GPLv2)

A Small Note on Terms

Free and Open Source Software (FOSS) (Also known as *Free/Libre Open Source Software*) is what we’ve been discussing. “Free software” refers to freedom and not to price.

Open Source Software is a term used by some software and technology companies. You may not have the right to use, modify, or redistribute the software or source code.

Additional Resources

Resources on learning:

- *How to ask good questions* by Julia Evans
- *How to teach yourself hard things* by Julia Evans

Technical resources:

- ArchWiki (“If the ArchWiki offered dating advice, no Linux user would be single." - Ex-SM of the OCF)
- The Linux System Administrator's Guide from The Linux Documentation Project
- The Debian Administrator’s Handbook by Raphaël Hertzog and Roland Mas

Fun reads about UNIX history and culture:

- In the Beginning was the Command Line by Neal Stephenson
- About the GNU Project by the Free Software Foundation
- Philosophy of the GNU Project by the Free Software Foundation

Today’s Schedule

- Course logistics and registration
- Shell Setup + interactive demo
- Lab introduction
- Work on lab!
- Open OH for the rest of the hour

Important Links and Resources

- Gradescope
  - Didn’t get added? Let me know right now!!
- Communications
  - Piazza
  - Slack: ocf.io/slack
  - Discord: ocf.io/discord
- These slides: on website, too lazy to make rewrite rules :P
- The lab: decal.ocf.io/labs/b1

Stuff to do

- Watch lecture before lab
- Show up or complete participation assignment on Gradescope
- Complete lab on Gradescope
- That's it!

Week 1-3: Digging into Linux

- This week: Terminal review
  - Go over some important commands
  - Automation with scripts

- Next week: Packages
  - What is a package, package managers, making your own packages

- Week 3: Setting up your computer and installing Linux
  - Virtual machines, file system partitioning, user setup

Installation

- Linux: you’re done :)
- Mac: you’re probably done, but make sure things work
- Windows:
  - Get WSL
  - Or Git Bash (git-scm.com)
  - ~~Or Cygwin (cygwin.com)~~ (pls no)
- iPad / phone / smartwatch / Samsung Smart Fridge ™
  - Terminal Emulator Apps
  - Maybe get another device

Demo Time!

- Licenses and open source in the wild
- Taking requests for shell stuff
- Some free lab tips + starters as a gift for showing up
- Server room tour (?)
- Q&A

vacuum